VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Metody umělé inteligence pro hru Starcraft

Methods of Artificial Intelligence for the Game StarCraft

2016 Vladimír Jaroň

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Vladimír Jaroň**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Metody umělé inteligence pro hru StarCraft
Methods of Artificial Intelligence for the Game StarCraft

Jazyk vypracování: čeština

Zásady pro vypracování:

SSCAI je studentská soutěž metod umělé inteligence pro hru StarCraft pořádaná Univerzitou Komenského v Bratislavě. Hra StarCraft obsahuje tři rasy lišící se zvolenou strategií. Podle pravidel soutěže se metoda umělé inteligence musí zaměřit na jednu z těchto ras. Cílem bakalářské práce je navrhnout a implementovat metodu umělé inteligence pro rasu Zerg, která je specifická řízením velkého počtu slabších jednotek.

1. Proveďte rešerši metod umělé inteligence.
2. Seznamte se s pravidly hry StarCraft se zaměřením na danou rasu a popište strategie úspěšné pro boj s ostatními rasami.
3. Nastudujte rozhraní soutěže SSCAI.
4. Implementujte vybranou metodu umělé inteligence na základě nastudovaných poznatků o chování rasy Zerg.
5. Otestujte a porovnejte vaší implementaci s existujícími metodami přihlášenými do soutěže.

Seznam doporučené odborné literatury:

[1] RUSSELL, Stuart; NORVIG, Peter. AI a modern approach (3rd Edition). Learning, 2009.
[2] MILLINGTON, Ian; FUNGE, John. Artificial intelligence for games. CRC Press, 2009.
[3] Comenius University, Bratislava. Student StarCraft AI Tournament. 2014. url: http://www.sscaitournament.com/.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Roman Meca**

Datum zadání: 01.09.2014

Datum odevzdání: 15.07.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě: 15. 7. 2016

Vladimír Jaroň

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli a podporovali mě. Dále bych chtěl poděkovat vedoucímu práce Ing. Romanu Mecovi, který mi poskytl užitečné rady a tipy k vypracování práce.

Abstrakt

Cílem této práce bylo vytvoření umělé inteligence a její implementace do hry Starcraft:Broodwar. Při vytváření byly použity tyto algoritmy Case-based řízení, Bayesian model.Výsledná umělá inteligence je samostatně schopná se rozhodovat a přizpůsobovat se prostředí, nebo protivníkovi. Výsledek této práce slouží k seznámení různých metod umělé inteligence pro hru StarCraft:Broodwar.

Klíčová slova: bakalářská práce, Umělá Inteligence, StarCraft: Broodwar, real-time strategie, hry, bayesian model

Abstract

Goal of this bachelor thesis was create Artificial inteligence and implementation itself to game name StarCraft: Broodwar. During creating program has been used these algorithms Case-Based planning, Bayesian model. Final Artificial Inteligence is capable make decisions and adapt to enviroment or oponent. The result of this thesis serves to educate diferent kind of Artificial Intelligence methods for StarCraft: Broodwar.

Key Words: bachelor thesis, Artificial Intelligence, StarCraft: Broodwar, real-time strategy, games, bayesian model

Obsah

Seznam použitých zkratek a symbolů

RTS	–	Real-Time Strategy (Strategická hra odehrávající se v reálném čase).
vs	–	Versus (Kdo proti komu stojí).
AI	–	anglicky Artificial Intelligence (česky UI- Umělá Inteligence).
HP	–	anglicky Hit Points (reprezentace množství životů dané jednotky).
GUI	–	anglicky Graphical User Interface (česky GrafickéUživatelské Prostředí).
BWAPI	–	anglicky Brood War Application Programing Interface. Zprostředkovává metody pro vytvážení a upravování AI modulů pro hru StarCraft:Brood War.

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Cílem této práce bylo seznámení se s metodami umělé inteligence s následným vytvořením a implementováním do počítačové hry StarCraft:Broodwar. V dnešní době je umělá inteligence všude kolem nás a dokáže si jí představit každý z nás, kdo se setkal s nějakým zařízením, které se chová samostatně bez vnějších vlivů uživatele. Například vysavač, který jezdí po bytě bez jakéhokoli zásahu uživatele, tak i složitější mechanismy nacházejícího se v robotice. Dále jí můžeme najít v kamerovém systému, kdy je schopna samostatně podle vloženého obličeje najít pachatele trestného činu. Dále a v neposlední řadě se umělá inteligence nachází v počítačových hrách.

V předchozím odstavci jsem uvedl pár příkladu umělé inteligence, ale co vlastně umělá inteligence neboli AI je? AI je vědní obor informatiky zabývající se vytváření takových mechanizmů, které jsou schopny se rozhodovat na úrovni lidského vědění. Jelikož se bakalářská práce zaměřuje na AI pro videohru můžeme si uvést příklad z herního průmyslu. Vezmeme si hru například, kde máme za úkol vybudovat silnou vojenskou základnu. K tomuto potřebujeme nějaké suroviny a jednotky, které jsou určeny právě pro těžbu. AI tedy musí najít nejbližší možný zdroj surovin a přiřadit k němu dělníka. Dále se AI musí starat o výstavbu budov. Dále AI musí dohlížet na stav surovin, které zbývají k maximálnímu vytěžení. Pokud se tak stane, musí hledat nové naleziště. Z tohoto příkladu můžeme vidět, že AI i pro zdárně jednoduché úlohy musí řešit několik možných variant typu úkonu.

Tato práce je zaměřena na vytvoření umělé inteligence neboli AI pro videohru StarCraft: Broodwar. Jelikož jsem fanda video her a od doby, kdy jsem dostal svůj první počítač, mě zajímá jak je možné, že hra ví co dělat za jakýchkoli okolností. Jsem byl velice rád za toto téma. Nejen, že jsem nastudoval různé metodiky umělé inteligence, ale navíc jsem jednu takovou naprogramoval.

2 Real time strategické hry

Real time strategické (RTS) hry je označení pro hry, kde hráč plní úkoly za pomocí svých jednotek. To všechno se děje v reálném čase. Čas je většinou shodný s časovou linii hráče, nicméně délky trvání určitých akcí jsou zkráceny s ohledem na délku trvání. Hráč má většinou na začátku hry určité množství surovin, budov a jednotek, díky kterým se musí pokusit splnit cíl hry. Jak název napovídá, vše se děje v reálném čase jako je např. výroba, těžba, výstavba a další možné úkony. Hráč si musí vše strategicky naplánovat tak, aby byl schopen za co nejkratší čas se přiblížit k cíli hry. Každá jednotka může vykonávat v daný čas pouze jednu úlohu.
RTS hry jsou většinou rozděleny do dvou skupin:

- Válečné hry
- Simulační strategie

Válečné hry
Hra je orientována na souboj mezi hráčem a soupeřem. Hráč se soustředí především na vybudování útočných jednotek, nebo obranu své základny. Mapa bývá většinou zakryta a hráč si jí musí sám prozkoumat. Mezi tyto hry můžeme zařadit:
Cold War Conflicts
Cold War Conflicts je RTS hra vsazená do období studené války roku 1950-1973. Ve čtyřech kampaních hráč přebere kontrolu nad vojenskou sílou různých národů (USA, Korea, Velká Británie atd.) Hra je tzv. nelineární, úspěšnost mise, nebo její selhání ovlivní další postup hry například: změna výchozího bodu pro další misi, nebo síla jednotek, kterou hráč bude disponovat. Hráč má na začátku mise pouze určitý počet jednotek a zásob. Hra je zaměřena především na vojenskou strategii a taktiku namísto těžby a výstavby základny.
Total War: Rome II
Cílem této hry je dobít různé území. Hráč si ze začátku hry vybere svou vlastí starověkou frakci a stane se jejím generálem. Nejen, že hráč ovládá vojenské jednotky, ale také se stará o vývoj měst ve svém regionu a o diplomacii mezi ostatními frakcemi. Dále si hráč vybírá různé technologie, které mu umožnují verbovat jednotky, nebo stavět různé budovy. Souboje jsou zde řešeny v reálném čase, kdy hráč ovládá své skupiny jednotek proti nepřátelské armádě. Každá jednotka má své vlastnosti, které se během bitvy mohou měnit:

- Morálka: jestliže jednotky stojí proti silnějším, nebo jednotka utržela velké ztráty tak morálka může spadnout a jednotka již není tak silná. Dokonce se může vzdát a utéct.
- Výdrž: u jednotlivých jednotek se výdrž může lišit, záleží na zbroji. Jestliže jednotky mají výdrž plnou, mohou se přesunovat rychle. Pokud ovšem výdrž klesne, jednotky se mohou přesouvat pouze chůzí.
- Množství munice: u střeleckých jednotek hraje velkou roli počet munice. Pokud jednotka vyčerpá svou munici může bojovat pouze na blízko.

Simulační strategie

Hry se především zabývají ekonomikou a prosperitou například firmy, nebo města. Příklady těchto her:

Railroad Tycon

Hráč je vsazen do pozice ředitele železniční společnosti. Cílem hry je vytvořit fungující železniční síť mezi městy. Nejedná se jen o přepravu obyvatel z města do měst, ale také o přepravu různého zboží. Hráč ze začátku disponuje určitým kapitálem. Při vytvoření první železniční stanice, má možnost koupi určitých vlakových souprav. Ze začátku hry má k dispozici pouze motorové vlaky, v pozdější fázi hry se odemykají i elektrické soupravy. Dále hráč má možnost rozšiřovat své železniční stanoviště jako například o: hotely, obchody, rozšíření depa atd.

Sim City

Cílem této hry je vybudovat své vlastní prosperující město. Hráč je v roli starosty města a ze začátku hry má omezené množství financí a prázdnou plochu připravenou k výstavbě města. Hráč má na výběr ze tří stavebních možností: civilní, obchodní a průmyslové budovy. Dále hráč musí zajistit elektriku a dostatek čisté vody. Podle toho jak jsou lidé ve městě spokojeni, se jeho město rozrůstá. Spokojenost lidí se odvíjí od výši daní po zamořenosti ovzduší.

2.1 Problémové oblasti pro vytváření Artificial Inteligence

Z pohledu vytváření umělé inteligence jsou RTS hry velice zajímavé. RTS hry poskytují velké množství problémů a úkolů pro vytváření AI. Díky velkému množství možností jak řešit tyto problémy žánr real-time strategií nabízí lidem zabývajících se grafovými algoritmy, nebo odborníkům na multiagentové systémy mnoho problému k řešení.

Z oblasti těchto problémů, na které během vytváření umělé inteligence můžeme narazit, si můžeme uvést některé z nich:

Hledání cesty

Díky velkému množství jednotek, které se vyskytují v RTS hrách, je hledání cesty jedna z nejčastějších volaných funkcí. Tudíž je vhodné pro ní hledat co nejefektivnější řešení. V některých hrách je volání funkce tak časté, tím se program spomaluje, a proto se musí hledat efektivnější způsob hledání.

Další možností kromě hledání nejkratší cesty může být hledání optimální cesty podle určitých kritérií. Jako je například hledání co nejbezpečnější cesty, nebo hledání takové cesty, aby se nenarušila formace jednotek.

Analýza terénu

Hlavním cílem analýzy terénu je zhodnocení a zmapovaní herní plochy. Toto nám může sloužit k v dalších oblastech, jako je například průzkum mapy, nebo rozdělení herní mapy na určité oblasti, se kterými pak můžeme pracovat samostatně.

Dalším úkolem analýzy může být vyhodnocené vhodného místa pro stavbu budov základny. To zejména platí pro určení polohy ke stavbě, tak aby neblokovala průchod jednotek a úspory

místa. Rozložení budov sloužících k obraně proti útočícím jednotkám, musí být ze strategického hlediska, taky rozumně uspořádány a rozloženy kolem základny.

Jiným využitím analýzy může být využití struktury terénu při bojích. Například využití úzkého průchodu, nebo vyvýšeného místa na mapě. Tato znalost terénu nám může pomoc k vítězství, protože i v menším počtu jednotek při boji v úzkém místě nám může pomoct proti velkému množství nepřátel a výsledek boje bude jiný než při střetu na otevřeném prostranství.

Aktivní zkoumání mapy

V RTS hrách je obvyklé, že hráč nemá přehled o celé hrací ploše. Ze začátku hry hráč má odkrytou pouze část, na které se nachází. K prozkoumání dalších částí mapy potřebuje hráč vyslat svou vlastní jednotku. Problém při průzkumu mapy může nastat, pokud hráč narazí na nepřítelovy jednotky. Zde hrozí ztráty vlastních jednotek, nebo prozrazení vlastní polohy.

Průzkum je velice důležitý a přináší nám různé informace nejen o tahu nepřítele a jeho momentální síly, ale také v objevení nových surovin. Informace z různých oblastí mapy nemusí mít stejnou hodnotu, tudíž je dobré si rozvrhnou, do které oblasti průzkumníka poslat.

Správa základny a surovin

Pro vytvoření silné armády musí hráč spoléhat na silně vybudovanou základnu. Základna nám slouží ke stavbě nových budov, které nám umožňují například zkoumat nové technologie, nebo k výrobě silnějších jednotek. Stavba základny, by měla podporovat určenou strategii, podle kterou si hráč zvolil. Dále je třeba počítat s obranou a podle toho rozmisťovat budovy, které jsou k tomuto určeny.

Dalším důležitým aspektem je správa surovin. Hráč má ze začátku hry určité množství surovin, se kterýma může pracovat. Hráč si pak musí zařídit těžbu nových surovin, aby si zajistil pevnou ekonomiku pro další vývoj hry.

2.2 Význam RTS pro Artificial Inteligence

2.2.1 Hra probíhá v real-time čase

Děj hry se odehrává v real-time čase a to ještě vcelku rychle, hry nebývají delší než hodina. Během tohoto úseku hráči musí být schopni postavit schopnou základnu, vybudovat velkou armádu a prozkoumat mapu, aby našli pozici svého nepřítele. Dále se musí postarat o obranu a o útok na nepřítele a to v co možná nejkratším časovém úseku, aby získal výhodu oproti ostatním hráčům.

2.2.2 Množství jednotek

Počet jednotek v RTS hrách jsou v desítkách někdy i ve stovkách současně se pohybujících na mapě. Artificial intelligence (AI) v těchto hrách musí umět vybírat takové akce, které nejsou časově náročné a tudíž rozhodování i poměrně velkého množství jednotek nezabere tolik času.

2.2.3 Aktivita více hráčů

Při hraní RTS musí hráč reagovat a být schopen odpovědět na úkon ostatních hráčů v daném okamžiku, jinak přijde o výhodu v dalším průběhu hry a může tedy ztratit náskok na soupeře. To znamená, že AI by mělo být schopno měnit své plány v závislosti na rozhodnutích svých protihráčů.

2.2.4 Znalost prostředí

RTS hry mají svou mapu zakrytou, hráč jí musí v první řadě prozkoumat, aby zjistil, kde se nachází nepřítelova základna. RTS hry mají tzv. Fog of war – hráči nestačí mít odkrytou mapu, ale aby mohl vidět nepřátelské jednotky, musí zde mít i svou vlastní jednotku. AI se tedy musí umět rozhodovat, kde a jakou jednotku může nechat, aby věděla o úmyslech a vývoji nepřátelských jednotek.

2.3 Hra Starcraft:Broodwar

Hra Starcraft:Broodwar je rozšíření do původní hry Starcraft. Starcraft je real-time strategie (RTS) vydána v roce 1998 společnostní Blizzard Entertainment a Saffire Entertainment. Hlavní příběh hry je válka mezi třemi galaktickými rasami: Protoss, Zerg, Terran. Protoss je rasa vyspělých humanoidních válečníků, Zerg je rasa hmyzu, něco jako mimozemšťané, kteří sdílí stejné myšlení, Terran je rasa lidských vězňů ze Země.

Obrázek 1: Ukázka ze hry StarCraft: Broodwar.

2.3.1 Princip hry

Na začátku každé hry hráč disponuje hlavní budovou a několika jednotkami tzv. Dronů, což jsou jednotky pro těžbu minerálů/plynu, nebo pro výstavbu budov. Pro výstavbu nebo verbování nových jednotek potřebujeme mít dostatečný počet natěžených minerálů nebo plynů. Plyn můžeme těžit po výstavbě rafinerie na určitém místě zvaném plynový gejzír. U každé startovní lokace je jeden gejzír a určité množství minerálů, pokud hráč minerály nebo plyn vytěží, musí najít nové naleziště. Dále si hráč musí hlídat velikost populace, která je definována počtem jednotek. Tato populace se zvedá u každé rasy jinou jednotkou nebo budovou. Každá rasa má jiné jednotky, budovy a technologie.

2.3.2 Rasa Zerg

Rasa Zerg je rychlá, agresivní rasa, která je především účinná v early game, tedy v počáteční fázi hry. Tato rasa se od ostatních liší především v produkci jednotek, které se vyvíjejí z larev. Tyto larvy se spawmují každých dvacet vteřin u každé hlavní budovy. Výhodou této rasy jsou náklady na výrobu jednotek, které jsou velmi nízké, proto strategií této rasy je rychlý útok s velkým množstvím jednotek. Nejčastější útočnou jednotkou je Zergling, který je velmi rychlý nejen pohybově, ale také v rychlosti útoku, avšak jeho poškození a odolnost vůči útokům jsou velmí nízké, proto spoléhá na množství a schopnost zvanou „burrow“. Burrow je vlastnost, která umožňuje se schovat nepřátelským jednotkám pod zemí. Jestliže nepřátelé nemají jednotky na odhalení je tato schopnost velkou výhodou. Nevýhoda této schopnosti je, že jednotka nemůže útočit nebo se pohybovat.[1]
Výhody:

- Nízká cena výroby jednotek
- Schopnotst regenerace jednotek a budov
- Schopnost burrow pozemních jednotek
- Jednotky se vyvíjejí z larev

Nevýhody:

- Menší síla jednotek
- Efektivita stavby budov
- Vědět, kdy je potřeba dronů pro stavbu, nebo jednotek na útok/obranu

2.3.3 Rasa Protoss

Rasa Protoss je velmi vyspělá, schopná a nejsilnější v této hře. Jednotky jsou velmi silné a mají velké množství životů. Nevýhodou je cena za výrobu jak jednotek, tak samostatných budov. Tato rasa dále potřebuje energii, která se vytváří v budově zvané Pylon. Při nedostatku energie rasa nemůže stavět ani vyrábět další jednotky. Toto je nevýhoda například proti rase Zerg, která má velké množství jednotek a neřeší cenu za výrobu. Jakmile Protoss překoná začátek hry a vybuduje si silnou ekonomiku, je těžké jej zastavit.[1]
Výhody:

- Vysoká síla jednotek
- Velká odolnost jednotek
- Regenerující se štít
- Obranná budova jak na pozemní, tak na vzdušné jednotky
- Silné speciální schopnosti

Nevýhody:

- Velké náklady na výrobu jednotek
- Velká spotřeba zásob na jednotku
- Potřeba Pylonů na výrobu energie

2.3.4 Rasa Terran

Rasa Terran díky průměrné ceně jednotek dokáže generovat armádu v relativně krátkém čase. Díky možnosti přesunu budov je rasa Terran považována za nejmobilnější rasou ve hře. Tato rasa má nejlepší rush vlastnost. Spolu s obranou proti rasám s podobným typem útoku, který je orientován na early rush je tato rasa nebezpečná.Útok jednotek je sice silný, ale výdrž velmi nízká. Díky lékařským jednotkám však síla útočných jednotek vyšší. Terran má dvě odvětví technologií. První je zaměřená na pěchotu (biologickou) a druhá na vozidla (mechanickou). Výzkumy těchto dvou odvětví nejsou propojeny, tudíž zaberou čas.[1]
Výhody:

- Velké poškození
- Útok na dálku
- Přemístění budov
- Zvýšena efektivita staveb

Nevýhody:

- Nízká odolnost jednotek
- Potřeba většího času přípravy před bojem
- Regenerace zdraví pouze za pomocí jednotek (medics)

2.4 Taktika rasy Zerg

Největší výhoda téhle rasy je nízká cena jednotek, které se vytvářejí z larev. Každá hlavní budova generuje tři larvy za dvacet sekund. Tudíž každých dvacet vteřin může vyrobit tři jednotky, pokud však máme dobře fungující ekonomiku. Jestliže se postaví budova Queen's Nest počet generovaných larev se zvýší na čtyři. Útočné jednotky jsou sice slabší v porovnání s ostatními rasami, ale Zergové spoléhají na množství útočících jednotek. Tím, že dokáží mít na začátku hry velké množství jednotek a zaútočí na nepřítele, mohou mu tak nabourat ekonomiku a zpomalit jeho výrobu jak jednotek, tak budov. Toto je veliká výhoda převážně v postupu do mid game. Zergové nemusí brát ohled na ztráty svých jednotek, tak mohou posílat na nepřítele jednotku za jednotkou a takzvaně jej harassmentovat od začátku hry tzn. obtěžovat nepřítele, především útoky na těžební drony, které slouží k nabourání ekonomiky. Nevýhodou jsou dražší náklady na výstavbu budov a obětování dronů pro stavbu, tato fakta mohou zbrzdit počáteční ekonomiku. Počáteční útok na ekonomiku nepřítele může vést v převahu Zergů. Pokud Zergové přežijí do mid-game, prostřední část hry kdy hráči již mají připravené základní jednotky a budovy a připravují se na samotný boj, mají možnost dohnat, ba dokonce předehnat ekonomiku dalších protivníků. Levnější cena hlavní budovy dovoluje větší množství expanzí a tím si zajistí silnější a stabilnější ekonomiku.[1]

2.4.1 Zerg vs Protoss

Zergové mají o dost větší množství jednotek oproti menší armádě, ale zato odolnější. To Zergům poskytuje rozložení množství sil do menších jednotek a útočit na více místech najednou. Tím mohou způsobit větší škody jak na jednotkách, tak na celé ekonomice. Pokud ekonomika Zergů funguje a v early game Protoss obdržel velké ztráty, je velmi malá šance, že by Protoss obrátil hru ve svůj prospěch.[1]

2.4.2 Zerg vs Terran

Zergové proti Protoss čelí velmi vážné hrozbě. Armáda z mariňáků a lékařů je vážná hrozba nejen pro Zergy. Tato kombinace může nadělat veliké škody s velmi nízkými ztrátami. Naštěstí Zergové mají jednotku Lurker, která je především určena k ničení velkých skupin armád. Tato jednotka těží z vlastnosti burrow a vyčkává, až je nepřítel přímo nad ní a udělí mu velké množství poškození. V kombinaci s ostatními jednotkami je velmi těžké Lurkery detekovat a zaútočit na ně.[1]

2.4.3 Zerg vs Zerg

Když Zerg čelí stejné rase, vývoj hry je předvídatelný. Hra většinou mívá tři typy útočných jednotek: Zergling, Multalisks a Scourge. Toto je především proto, že Hydralisk není efektivní při boji proti kombinaci Zergling/Multalisk. Devouder jsou občas využívány, ale především pokud se jedná o delší hry. Velmi efektivní může být nydus canal, kterým se jednotky rychleji mohou přesouvat po mapě. Princip nydus canalu spočívá v tom, že vytvoří tunel ze dvou bran, kdekoli na mapě a jednotky je pak mohou využívat k rychlejšímu cestování. Tato taktika avšak potřebuje rychlé odhalení nepřítelovy základny.[1]

2.5 Zvolená taktika

Podle výše uvedených vlastností rasy Zerg jsem zvolil early-game taktiku, zaměřenou na harassmentování nepřítele v co nejrychlejším čase, abych mu naboural ekonomiku a znepříjemnil postup do mid-game. Pro skautování mapy a objevení nepřátelské základny jsem nasadil útočnou jednotku Zergling ve větším počtu, v mém případě šest jednotek. Skauti prozkoumávají danou mapu podle určitým výchozích bodů, až po nalezení nepřítele. Po objevení základny jsem rozdělil nepřítelovy síly do čtyř kategorií:

- Útočné jednotky – pozemní jednotky, které mohou útočit
- Útočné budovy – budov, které poskytují obranu
- Dělníci – jednotky pro těžbu minerálů
- Ostatní budovy – zbytek budov

Útočné jednotky se skládají především ze Zergling. V pozdější části hry, jestliže je postavena budova Hydralisk den, začnou se vyrábět jednotky zvané Hydralisk, který se po dokončení Lurker den vyvine do Lurker, který je silnější a jeho hlavní výhodou je útok z podzemí. Pokud nepřítel disponuje budovami na útok je útočná síla nejdříe zaměřena na tyto budovy. Dále se pak soustředím na útočné jednotky, které mohou udílet poškození. Pokud jsou tyt dvě hrozby zneškodněny, útočné jednotky se zaměří na těžební drony a zbytek budov. Vývoj technologií jsem zaměřil pouze pro vylepšení útoku a rychlosti pohybu pro útočné jednotky. Jediná výjimka je vývoj technologie burrow, která umožňuje jednotkám se zakopat a tím se stát neviditelnými pro nepřátelské jednotky a získat výhodu při útoku. O obranu základny proti pozemním jednotkám se starají budovy suken colony, jenž jsou vylepšením budov creep colony. Budova creep colony je důležitá v další věci, a to v rozšíření místa pro výstavbu dalších budov. Rasa Zerg nemůže stavět budovy volně po zemi, ale musí mít svůj vlastní biologický podklad. Ten je získán na začátku hlavní budovou nebo dalším rozšířením právě creep colony.

3 Umělá inteligence

Definicí umělé inteligence je mnoho. Žádná z nich není obecně přijata. Uvedeme si některé z nich. Umělá inteligence je věda o vytváření strojů nebo systémů, které budou při řešení určitého úkolu užívat takového postupu, který – kdyby ho dělal člověk – bychom považovali za projev jeho inteligence. (Minsky, 1967) Umělá inteligence se zabývá tím, jak počítačově řešit úlohy, které dnes zatím zvládají lidé lépe. (Rich, Knight, 1991)[2] Následující tabulka č.1 zobrazujerozdělení definic umělé inteligence do čtyř kategorií.

Myslet lidsky	**Myslet racionálně**
„Nová vzrušující snaha nechat počítače myslet...“stoje s mozkem“, v plném znění“ (Haugeland, 1985)	„Studie duševních schopností, pomocí použití výpočetních modelů“ (Charniak and McDermott, 1985)
„[Automatizace] aktivity, které spojujeme s lidským myšlením, jako jsou rozhodování, řešení problémů, učení ...“ (Bellman, 1978)	„Studie výpočtů, které umožňují vnímat, přemýšlet a jednat.“ (Winston, 1992)
Jednat lidsky	**Jednat racionálně**
„Umění vytváření strojů, jenž provádějí funkce , které využívají inteligenci lidí.“ (Kurzweil, 1990)	„Výpočetní inteligence je věda o vytváření inteligentních agentů.“ (Poole et al., 1992)
„Studie o vytváření počítačů, které dělají věci, ve kterých jsou lidé lepší.“ (Rich a Knight, 1991)	„AI ...se zabývá inteligentním chováním artefaktů.“ (Nilsson, 1998)

Tabulka 1: Základní definice AI.[2]

3.1 Znaky umělé inteligence

Jednat lidsky: Turningův test

Tenhle test vymyslel Alan Turning (1950), byl navržen tak, aby poskytnul uspokojivou operační definici inteligence. Turning definoval inteligentní chování, jako schopnost dosáhnou lidské úrovně ve všech poznávacích otázkách, tak aby oklamal vyšetřovatele. Test spočíval v tom, že v jedné místnosti sedí člověk a skrz terminál pokládá písemné otázky. Na tyto otázky odpovídají v druhé místnosti člověk a počítač. Otázky jsou rozdělovány mezi člověka a stroj, pokud tázající člověk nedokáže rozeznat odpověď člověka a stroje, je tento stroj možné považovat za inteligentní. [2] K tomu počítač potřebuje následující schopnosti:

- Zpracování jazyka - schopnost komunikovat v Angličtině (nebo v jiném lidském jazyce)
- Reprezentace znalostí - ukládání informací před nebo v průběhu výslechu

- Automatizovaná úvaha - využití uložených informací k zodpovězení otázky a vyhotovení nového závěru
- Mechanické učení - přizpůsobovat se novým okolnostem, odhalovat a extrapolovat vzory

Myslet lidsky
Jestli chceme říct, že daný program myslí jako člověk, musíme mít nějaké ponětí o tom, jak člověk myslí.[2] Toho můžeme dosáhnout třemi způsoby:

- sebepoznání – pokusit se zachytit své vlastní myšlenky
- psychologický test – pozorování člověka v akci
- zobrazení mozku – pozorování přímo mozku v akci

Myslet racionálně Od roku 1965 existují programy, které jsou schopny, v podstatě vyřešit jakýkoli řešitelný problém popsaný v logické notaci. Teorie logiky v rámci umělé inteligence doufá ve vybudování takových programů, které by vytvořily inteligentní systém.[2] Vytvoření těchto programů komplikují dva hlavní problémy:

- Není jednoduché vzít neformální znalosti a převést je do formálních podmínek požadovaných v logických notacích, zvláště když jsou znalosti menší než 100%
- Je zde velký rozdíl v řešení problému teoreticky a řešení jej v praxi

Jednat racionálně: Agent
Za agenta je považováno něco, co jedná. Všechny počítačové programy něco dělají, avšak od agenta je očekáváno více věcí.[2] Jako například:

- Jednat autonomně
- Vnímat své prostředí
- Adaptovat se na změny
- Vytvářet a plnit úkoly
- vytrvat v delším časovém úseku

3.2 Metody pro řízení Umělé Inteligence

Zaměříme se přímo na problematiku hry Starcraft. U hry Starcraft můžeme rozhodování rozdělit na dva hlavní proudy, a to na taktické a strategické rozhodování. Tabulka č.2 zobrazuje AI techniky použité ve hře Starcraf.

Taktické rozhodování	Strategické rozhodování a rozpoznávací plán
Reinforcement Learning	Case-Based Planning
Game-Tree Search	Hierarchical Planning
Bayesian Models	Goal-Driven Autonomy
Neural Networks	State Space Planning
	Evolutionary Algorithms
	Cognitive Architectures
	Deductive Reasoning
	Probabilistic Reasoning
	Case-Based Reasoning

Tabulka 2: AI techniky použité ve hře Starcraft[3]

3.2.1 Taktické rozhodování

Ovládání jednotek v krátkém časovém úseku a rozhodování se s konkrétními jednotkami často nazýváme micromanagement. Tento způsob rozhodování je vhodný především při útoku. Správné rozhodnutí, kdy a kde zaútočit, může rozložit taktiku a ekonomiku protihráče a přinést tak vítězství.

3.2.2 Bayesian programming

Představíme si Bayesian programs (BP) jako formalismus, který může být použit pro popsání veškerých typů Bayesianova modelu, který zahrnuje Bayesian networks a Bayesian maps, ekvivalent pravděpodobnostnímu grafu. BP má dvě hlavní části, popis jak vypočítat společnou distribuci a otázka(y), které budou pokládány.

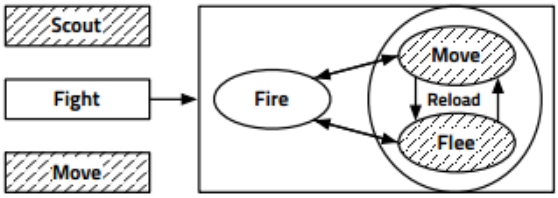
Popis obsahuje informaci o aktuálních proměnných a objasnění jejich závislostí pomocí rozdělení jejich vzájemných distribucí s existujícími známými daty. Forma každého funkčního produktu nám specifikuje jak vypočítat jejich distribuci. Dokonce parametrizované formy (zákony, nebo pravděpodobnostní tabulky mohou být brány z dat), nebo rekurzivní otázky z dalšího Bayesian programu. Odpověď na otázku se počítá z distribuce P(Hledaný/Známý), kdy Hledaný a Známý jsou dvě disjunktní podmnožiny proměnných. Obecně Bayesian interface je prakticky neřešitelný, ale podmíněné nezávislé hypotézy a omezení (stavy v popisu) často zjednoduší model. Také je zde velmi dobře známá přibližovací technika například Monte Carlo metody a různé variace Bayes.

Použijeme Bayesian programování jako alternativu k logice, která transformuje nekompletní informace z okolí do neznáma. V našem případě management jednotek, kde máme intenzivní nejistotu. Na místo pokládání otázek jako například: Kde se bude naše jednotka nacházet další frame? Nebo 10 snímků později. Náš model je založený na hrubém odhadu, který není brán jako fakt. Znalost odpovědi na tyto otázky bude požadována komunikace mezi vlastními jednotkami a striktně se držet plánu.[4]

Naše jednotky jsou jednoduše definované hierarchické konečné stavové automaty (stavy můžou být reprezentovány jako režimy), které mohou skautovat, pohybovat se, útočit. Každý druh jednotky má vlastní čas přebíjení a prodlení mezi útoky. Režim útoku, by tedy mohl vypadat následovně: [4]

if canFire to t = selectTarget() in inRange(t) **then**
    attack(t)
else if needFlee() **than**
    flee()
else
    flightMove()
end if

Na obrázku č.2 můžeme vidět grafické znázornění.



Obrázek 2: Bayesian unit modal.[4]

3.2.3 Reinforcement Learning

Reinforcement Learning (RL) je oblast počítačového učení, ve které se agent učí, metodou pokus a omyl, při správné akci je agent odměněn. Tato metoda je složitá, jelikož agent nikdy neví, která akce je správná, nýbrž která odměna patří k jaké akci. Při více jednodušších iterací RL může najít lepší řešení daného problému. Implementace do nové domény, která vyžaduje pouze popis situace a možných akcí, by pak měla být relativně snadná.

RL byl aplikován do hry Starcraft programátory Shantia, Begue a Wieringem (2011). Algoritmus Sarsa pro řešení RL problémů, je použit pro učení kontroly jednotek v malých bojích. Využili umělé neuronové sítě k naučení očekávané odměny za útok nebo stažení se s konkrétní jednotkou v daném stavu a vybrat akci s nejlepší možnou odměnou ve hře.[3]

3.2.4 Monte-Carlo Planning

Monte-Carlo Planning (MCPlan) je mechanismus založený na simulaci a vytváří příklady možných výsledných výsledků. Po několika opakováních s různými vstupními parametry vyhodnotí nejlepší možný výsledek. Tato metoda je efektivní k řešení náhodných a nekompletních problémů s alternativními kroky, jako je například poker nebo šachy. Velkou výhodou MCPlanu je redukce nutnosti znalosti množství informací. MCPlan spoléhá na efektivitě vyhodnocení funkce.[5]
Zjednodušený pohled na MC simulaci

1. Načítání předdefinovaných plánů postupem do simulátoru
2. Simulování každého plánů (jednoho daného a dalšího náhodného), následného vyhodnocení a ohodnocení plánu
3. Výběr plánu s nejlepším výsledkem a znovu jej simulovat a vyhodnotit
4. Opakovat krok 3
5. Výběr plánu s největším průměrným výsledkem pro AI ve hře

3.2.5 Neuronové sítě

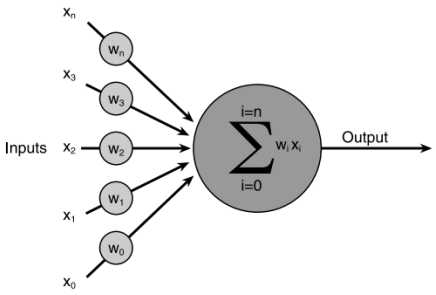
Neuronová síť je algoritmus, který má za vzor činnost lidského mozku. Jak je známo mozek je tvořen velkým množstvím vzájemně propojených buněk, které nazýváme neurony, jež spolu komunikují. Již od vzniku prvních počítačů se programátoři snaží vymyslet takový algoritmus, který bude napodobovat funkci lidského mozku. Princip neuronových sítí je v dnešní době implementován do řady dostupných rozhodovacích softwarů a v různých oborech lidské činnosti podává extrémně dobré výsledky.[6]

Princip sítě

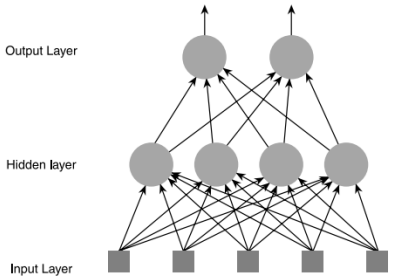
Softwarové neuronové sítě se inspirují biologickou neuronovou sítí, kde je základním stavebním kamenem neuronová buňka – neuron. Jednotlivé neurony jsou vzájemně propojeny spoji ohodnocenými vahami. Toto propojení a schopnost se adaptovat (učit se) na základě trénovačích vzorů v datech, dává neuronové síti nové široké množství v oblasti analýzy dat. [6]

Jak již bylo zmíněno největší předností je schopnost učit se, tedy zapamatovat si kombinace, které vedly k požadovanému výsledku a u nových vstupů se potom obracet na paměť a na základě zkušeností odhadovat nový výsledek. V tomto případě je myšlena generalizace, která je další výhodou těchto sítí. Jednoduše řečeno, jde o přiměřenou dovednost správně zareagovat i na vstupy, které nebyly součástí trénovačích dat a vyvodit z nich obecné závěry o datech.[6]

Další vlastností neuronových sítí je řešení silně nelineárních úloh. Využití neuronových sítí má význam tam, kde selhávají „klasické“ modely jako je regrese. V některých případech se stává, že není možné najít jednoduchou matematickou funkci, která vhodně postihla všechny vlivy, které variabilitu sledované proměnné ovlivňují. Potom je neuronová síť vhodná.[6]



Obrázek 3: Model jednoho neuronu.[7]



Obrázek 4: Propojení neuronů.[7]

3.2.6 Strategické Rozhodování

Strategické rozhodování, nebo také známé jako macromanagement, slouží ke splňování určitých cílů, především s cílem maximalizovat ekonomickou stránku hry. Cílem je vytvořit systém, který se dokáže inteligentně rozhodovat na strategické úrovni real time strategické hry. Tyto systémy jsou schopné přijímat sekvence akcí v určitých situacích, tak aby bylo dosaženo určitých cílů. Dosažení takových cílů je v RTS hrách vcelku komplikované a to díky Fog of War, nebo velkému množství stavů, akcí a simultánně probíhajících cílů. Vědci zabývající se umělou inteligencí doufají k vytvoření takového AI, které by dokázalo hrát na lidské úrovni.

3.2.7 Hierarchical Planning

Rozdělení problémů hierarchicky, je plánovací systém schopný se vypořádat s částmi situace odděleně na různých úrovních abstrakce. Tyto úrovně nám umožňují redukci složitosti situace, ale také vytvářejí nové potencionální problémy mezi vytvořenými úrovněmi. Hierarchická plán dobře mapuje cíle a podcíle v RTS hrách, od nejvyššího cíle, což je vyhrát hru, tak až po nižší úrovně, které mapuje v průběhu hry.

Někteří výzkumníci formalizovali hierarchii do dobře definované struktury zvané Hierarchical Task Network (HTN), který obsahuje úlohy, jejich interpretaci a metody pro jejich splnění. U složitějších úloh může HTN rozložit úlohu do sekvenci dílčích, jednodušších pod úloh, které mohou být reprezentovány jako dílčí kroky.

HTN bylo požito pro strategické rozhodování v RTS hrách, ale ne pro hru StarCraft. Munoz-Avila and Aha se v 2004 zaměřili na vysvětlování, že HTN plánovač je schopný provádět lidskou posloupnost chování, nebo provádět rozumné kroky v kontextu RTS hrách. Laagland v roce 2008 implementoval a otestoval agenta schopného hrát open source hru zvanou Spring použitím ručně naprogramovaného HTN. HTN umožnil agentovi dynamicky reagovat na problémy, jako jsou například přestavbu budovy, která byla zničena, nebo těžbu potřebných surovin určitého typu pokud byla potřeba. Použitím vybalancované strategie HTN agent byl schopný porážet vestavěnou AI ve hře Spring.[3]

3.2.8 Case-based Planning

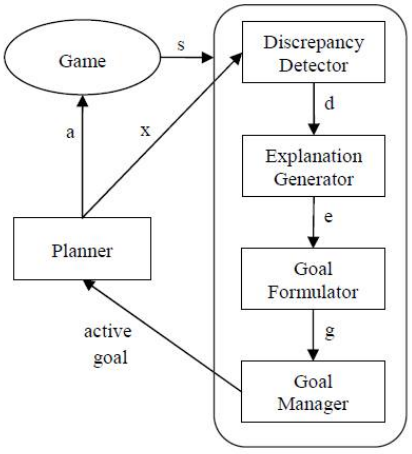
Case-Based Planning (CBS) je plánovací technika, která nachází podobné situace, ze kterých vykreslí potencionální řešení dané situace. V případě CBS systémů jsou nalezené výsledky souborem potenciálního plánu, nebo dílčích záměrů, které by mohly být efektivní pro danou situaci.

Prvním nasazením CBS do RTS her bylo v roce 2005 programátory Aha, Molineaux a Ponsen, vytvořili systém, který byl rozšířením „Dynamického skriptování“ Ponseonova konceptu vybírání taktiky a strategie založené na konkrétní situaci. Použitím této techniky byl jejich systém schopen hrát proti dynamickému protihráči. Zredukovali složitost stavů a akci tak že, vytvořili build order kombinovaný s malou sadou funkcí. Pro každý stav je generována sada taktik. Toto sady taktik systému dovolují zlepšování odhadu výkonosti jednotlivých taktik za každé situace během několika her a dokonce se naučit postupně porážet testované protivníkovy skripty.[3]

3.2.9 Goal-Driven Autonomy

Goal-Driven Autonomy (GDA) slouží k vytvoření takového agenta, který je schopný reagovat na neočekávané chyby, jenž mohou nastat při vytváření plánů. GDA přistupuje k tomuto problému tak, že agent má povědomí o svých cílech. Výzkum je založený na Coxově tvrzení, že agent má povědomí sám o sobě, tak dobře jak o okolním světě.[8] Zjednodušený GDA se skládá ze čtyř komponentů:

- Detekce rozporu
- Generátor objasnění
- Formulace cílů
- Manažer cílů



Obrázek 5: Componenty v zjednodušeném Goal-Driven Autonomy modelu.[8]

Detekce rozporu
Detektor kontroluje zda výsledný stav je schodný s očekávajícím se stavem. Jestliže je detekována nesrovnalost mezi očekávaným a výsledným stavem, agent vytvoří nesrovnalost. Tato nesrovnalost je pak dále poslána do generátoru objasnění.[8] Náš systém generuje nesrovnalosti v závislosti na následující typy událost:

- Nesrovnalost jednotek: oponent vytvoří novou jednotku
- Nesrovnalost staveb: oponent postaví novou budovu
- Nesrovnalost v rozvoji: oponent se rozvine
- Nesrovnalost útoku: oponent zaútočí
- Nesrovnalsot v síle: rozdíl mezi sílou oponenta a agenta

Generátor objasnění
Generátor objasnění na zákaldě souboru pravidel chování vygeneruje vysvětlení.[8] Můžeme vygenerovat jedno z následujících objasnění:

- Oponent se učí
- Oponent vyrábí letecké jednotky
- Oponent vyrábí neviditelné jednotky
- Oponent vyrábí detekční jednotky
- Oponent expanduje
- Agent má výhodu itemOponent má výhodu

Formulace cílů
Formulátor cílů zde vytváří stále nové cíle, v závislosti na generátoru objasnění.[8] Například jeden nebo více následujících cílů:

- Strategie: výběr strategie k vykonání
- Rozšíření: vybudování rozšíření a trénink jednotek na těžbu/stavbu
- Útok: útok na protivníka se všemi jednotkami
- Ústup: poslání všech útočných jednotek zpět na zákaldnu

Manažer cílů
Manažer cílů je zodpovědný za zvolení a splnění daného cíle. Přesto, že je náš systém schopný zpracovávat více cílů souběžně, aktivní cíl může být jen jeden na strategické, ekonomické a taktické úrovni.[8]

4 Implementace Uměle Inteligence

4.1 BWAPI

Brood War Application Programing Interface (BWAPI) je volně dostupný open source C++ framwork pro vytváření AI modulů pro hru StarCraftBrood War. Prostřednictvím BWAPI programátor dostává informace o hráčích či jednotkách. Dále také může dávat příkazy, což otevírá možnosti pro vytvoření vlastního micro a macro algoritmu.

4.1.1 Funkce BWAPI

BWAPI odhaluje pouze viditelné části herních stavů. Informace ohledně jednotek, které jsou ve fog of war jsou zamítnuté. Toto umožňuje programátorovi psát kompletně non-cheating AI, které musí plánovat budoucí operace pouze s částečnými informacemi. Navíc BWAPI zakazuje Starcarf GUI (Graphical User Interface), které nedovoluje uživateli přímo zasahovat do hry. Uživatel je pouze v roli diváka. Nicméně na začátku zápasu AI modul dovoluje jednu nebo dvě možnosti cheatu., což zvýší funkcionalitu BWAPI. První z nich dovoluje zobrazení informací jednotek, které jsou právě zakryté ve fog of war. Druhá z nich povoluje GUI, tudíž uživatel může hrát souběžně s AI modulem.

- Psaní konkurenčního AI pro StarCraft:BroodWar pomocí kontroly jednotlivých jednotek
- Čtení všech dostupných aspektů hry
- Analýza záznamů snímek po snímku a zkoumání požitých strategií
- Celkové informace o typu jednotky, vylepšeních, technologiích, zbraních
- Studování a výzkum real-time AI algoritmu ve velkém komerčním RTS prostředí

4.1.2 Třídy v BWAPI

BWAPI obsahuje několik tříd, které využíváme při tvorbě vlastního AI. Výstupem programu je pak AIModule.dll soubor, který se vloží do složky se StarCraft hrou. Po následném spuštěním hry nám tento soubor nahraje všechny použité dll moduly s AI a přidá je do hry. Výčet nejpoužívanějších tříd v BWAPI:[9]
AIModule: třída, která obsahuje eventy ve hře jako například:

- OnStart() - volá se jednou a to pouze na začátku, pro inicializaci
- OnEnd() - volá se jednou a to na konci hry
- OnFrame() - volá se jednou za každý jeden vykonaný časový úsek hry StarCraft
- OnUnitDiscover() - volá se pokaždé, když se objeví nová jednotka

- OnUnitDestroy() - volá se pokud je jednotka zničena
  Z této třídy je hlavně důležitá metoda OnFrame() kterou využívám pro hlavní smyčku (loop) programu. V OnStart() metodě si nadefinujeme hlavního agenta a vše potřebné pro třídu MapInfo, ve které si ukládám veškeré potřebné informace o mapě, jako je například pozice startovních lokací, pozice minerálů na mapě, svou vlastní a pozici objeveného nepřítele.

Game: abstraktní třída obsahující všechny možné herní stavy Starcftu. Tato třída obsahuje například informace ohledně všech:

- Jendotek
- Hráčích
- Terénu
- Regionech
- Fog of War

Unit: třída, která obstarává veškeré informace o individuální jednotce, také však obstarává příkazy pro danou jednotku, například:

- Zjištění typu jednotky
- Zjištění pozice jednotky
- Příkaz pro pohyb
- Rozkaz pro útok
  Tato třída je nejpoužívatelnější. Tuto třídu používám při každém příkazu pro jednotku. Dále ji využívám i pro kontrolu typu jednotky, nebo pro stav, ve kterém se jednotka nachází, například zda útočí, neprovádí žádnou akci, nebo je v pohybu.

UnitType: třída, sloužící pro zjištění konkrétních informací o dané jednotce, jako je například:

- Cena za výrobu jednotky
- Čas výstavby jednotky
- zjištění HP
- Typ zbraně používané danou jednotkou
  Tato třída mi napomáhá například u kontroly, zda se budova začala stavět a to tak, že si ověřuji množství hit pointu (HP). Dále také slouží ke zjištění množství potřebných surovin ke stavbě budovy nebo verbování jednotky.

WeaponType: třída, která nám podává informace o identifikaci zbraně, kterou používá daná jednotka. Některé typy zbraní mohou být vylepšeny.

- Dosah zbraně
- Čas přebíjení
- Poškození
  Třída, která mi pomáhá jak při útoku tak obraně a to především při výpočtu palebné síly nepřítele, nebo mé. A to tak, že díky informaci o velikosti poškození dané zbraně, kterou jednotka disponuje.

4.2 Macromanagement

Macromanagement se ve hře stará o výrobu jednotek a budov, které jsou důležité v dané fázi hry. Macromanagement je založen na principu Hierarchického plánování, které je zmíněné v podkapitole 3.2. O stavbu budov se stará ConstructAgent, který má za úkol vybírat budovy, které se mají postavit. Další funkcí, kterou má ConstructAgent je kontrola možnosti updatovat budovy do dalšího vývojového stupně.

O verbování jednotek se stará UnitAgent. Ten má za úkol kontrolovat, zda není požadováno vyrobit určité jednotky. Jednotky jsou rozděleny podle role na pracovní a útočné. Jakmile požadujeme verbovat jednotky, nastaví se se stop stav, což je booleovská hodnota, na true. Tento stav má za úkol zastavit všechny další úkony, které se týkají výstavby, a dá prioritu verbování dané jednotky. Po té co se jednotka vyrábí a není za potřebí dalších jednotek, stop stav se vrátí zpět na false a může pokračovat výstavba budov.

Další funkce macromanagementu je starost o těžbu minerálů nebo plynu. Těžba surovin je velice důležitá pro následnou stavbu základny a verbování jednotek. Při nedostatku surovin je zbrzděn postup do dalších fází hry, což může mít za následek prohru. Čím více má hráč k dispozici surovin tím efektivnější a rychlejší je postup.

4.2.1 Těžba surovin

O těžbu minerálů se starají dronové. Na začátku hry mám k dispozici 4 drony, kteří okamžitě po spuštění hry jdou těžit minerály k nejbližšímu zdroji s minerály. Každý dron má určený svůj vlastní zdroj minerálů, aby nedocházelo ke zdržení těžby. Například jakmile u jednoho zdroje těží dva dělníci, dochází k vyhýbání dvou dronů na cestě se surovinami k hlavní budově a to má za následek výběr delší cesty k vyhnutí a tím pádem k prodlení doručení. Toto je v mém případě řešeno tím, že každý dělník jde těžit ke zdroji minerálů kde se nenachází žádný další dron. V mém případě se o těžbu minerálů stará šest dronů. Každý dron unese za jednu cestu, která trvá 8 sekund 8 jednotek minerálů. Takže za jednu minutu disponuji 360 nových minerálů.

Po dokončení budovy na extrakci plynů z plynového gejzíru se začíná těžit i plyn. Silnější jednotky, nebo pokročilejší budovy potřebují k výstavbě nejen minerály, ale také tento plyn.

Tak samo, jako u těžby minerálů každý dron unese 8 jednotek plynů. V mém případě se o těžbu plynů starají dvě jednotky dronů. Počet nově natěžených plynů za jednu minutu je 120 kusů plynu.

4.2.2 Verbování jednotek

Jak již bylo zmíněno o verbování jednotek se stará Unit Agent. Začátek hry je zaměřený na verbování jednotek na těžbu minerálu. Dokud počet jednotek na těžbu není šest. Tento počet nám zajistí dostatečný počet natěžených minerálů v průběhu hry. Množství jednotek pro těžbu minerálů, nebo plynů se během hry může lišit v závislosti na rase neřpítele. Například proti rase Terran je počet dělníku po výstavbě nové základní budovy zvané Hatchery navýšen. Pokud je dosaženo množství natěžených minerálů ceně výroby dělníka a je potřeba další nový dělník tak se dá příkaz pro výrobu. Jestliže je volná larva může se dělník začít vyvíjet. Každý dělník má svůj čas na výrobu, jakmile je výroba hotova, dělník okamžitě jde těžit minerály. Po dovršení daného počtu pracovníků pro těžbu minerálů, se výroba dělníků zastaví a čeká se na další pokyny. Po dokončení rafinerie pro těžbu plynu se začnou vyvíjet dělníci na těžbu plynu.

Dále se musí kontrolovat populace. Ta se počítá podle vytvořených jednotek, počítají se veškeré jednotky krom budov. Populace u rasy Zerg se i odčítá. Jelikož budovy se vytváří z dronů, který zahyne a z něj se stane daná budova, tím se populace sníží o jednu. Jakmile aktuální populace se blíží k maximální, dá se příkaz pro výstavby jednotky Overloard. Tato jednotka nám zajistí navýšení maximální populace o osm. Ukázka kódu 1 pro verbování jednotek přiložen v příloze.

4.2.3 Výstavba budov

Výstavbu budov má na starosti Construct Agent. V němž je build order (posloupnost budov), které se mají postavit. Tato posloupnost se mění v závisloti na druhu rase, proti které můj bot nastoupí. V každém cyklu se kontroluje, která z budov má být postavena, je postavena, nebo se začala stavět. Jestliže je potřeba postavit určitou budovu a máme k dispozici dostatečný počet minerálů určený ke stavbě dané budovy, dá se rozkaz ke stavbě. Jakmile se budova začne stavět, posuneme se v build orderu dále na další budovu. Výstavba je závislá na verbování jednotek. Jestliže je potřeba verbovat útočné jednotky, výstavba budov je pozastavena a čeká na dokončení verbování.

Dále se Construct Agent zabývá updatem. Updaty budov a technologií jsou velice důležité pro další fázi hry. Důležitý je především vývoj další úrovně budovy Creep colony na Suken colony. Suken colony nám zajišťuje pozemní obranu proti nepřátelským jednotkám. Tudíž jakmile je postaven Creep colony okamžitě se zařizuje její update. Technologie, které jsou zkoumány, jak jsem již říkal, jsem zaměřil na zvýšení útočné síly jednotek. Jakmile je vyslána jednotka na skautování začne se vyvíjet technologie zvaná „Burrow“. Dále po dokončení stavby budovy Evolution chamber je zkoumána technologie Zerg Meele Attacks, která nám zvyšuje poškození

u zerglingů a hydralisků. Jakmile je dosažen počet minerálů a plynů na navýšení úrovně hlavní budovy a budova Evolution chambre je postavena začne se s upgradem hlavní budovy na Hive. Tento další stupeň hlavní budovy nám zajišťuje stavbu další pokročilých budov.

Při útoku nepřítele a ztrátě některé budovy se vytvoří nový list, do kterého se přidávají zničené budovy. Tento list nám slouží k obnovení budov, které jsem měl postavené. Původní build order se nezmění, jen se pozastaví a vyčkává na dostavění všech zničených budov.

4.3 Micromanagement

Micromanagement je zaměřený na skauting a především na samotný souboj. Skauting je prováděn za pomocí „Zergling“ jednotek. Znalost mapy je velmi dobrá zbraň k získávání informací nejen o nepřítelových jednotek a jeho vývoje, ale také k objevení nových těžebních pozic minerálů a plynů. K útoku jednotek je použit Bayessian model, který je uveden v podkapitole 3.2. Podle obrázku č.2 můžeme vidět taktiku tohoto modelu. Model je rozdělen do tří akcí:

- Boj.
- Pohyb.
- Skauting.

4.3.1 Skauting

Skauting neboli prohledávání mapy je prováděn pomocí „Zergling“ jednotek. Na začátku hry jsou známé možné počáteční pozice, na kterých se nepřítel může nacházet. Jakmile jsou k dispozici Zerglingové začne se provádět průzkum. Jednotky se rozdělí a každá jednotka jde prozkoumat jednu pozici. Jakmile je nepřítel objeven, jeho poloha je uložena a všechny ostatní jednotky začnou mířit k nepřátelské pozici. Výchozí polohy jsou získány pomoci BWAPI a dále je třídím od nejbližší možné. Skauti se na místě, kde objevili nepřítele, rozhodnou zda zaútočí nebo se stáhnou. Pokud se mají stáhnout, mohou to provést dvěma způsoby: vrátit se na bezpečnou pozici, nebo pokud je vyzkoumaná technologie „burrow“, tak díky ní se zakopat a vyčkat na posily. Jestliže dobijeme základnu nepřítele, ale nepřítel ještě není úplně zničen a má postavenou základnu mimo výchozí pozici, tak veškeré jednotky se rozprchnou po celé mapě a začnou ho hledat. Jakmile najdou nějaký cíl, tak všechny jednotky se k němu sejdou a zaútočí. Tento proces se opakuje do té doby, dokud nepřítel není zcela zničen. Ukázka kódu 2 pro skautování v příloze.

4.3.2 Pohyb

Pokud jednotka skautů objeví nepřítele, uloží se pozice nepřítele do třídy MapInfo. Jakmile jednotka narazí na první nepřátelskou jednotku, vezme se pozice skautů a podle známých výchozích poloh se vypočte nejbližší možná pozice a ta se uloží. Následná pozice je pak určena pro ostatní útočné jednotky, které ji využijí při pohybu k nepřítelovi. Pokud se bavíme o jednotce,

která provádí skauting a dojde na určenou pozici a nepřítel není nalezen, putuje na další možnou výchozí pozici.

4.3.3 Boj

Samostatný souboj se odvíjí z predikce možného vítězství, které se počítá z celkového poškození všech jednotek v oblasti, berou se i budovy, které slouží jak na pozemní tak na vzdušnou obranu. Podle typu jednotek se vypočítá poškození, které udílí daná jednotka. Každá jednotka má k dispozici jinou zbraň a ta udílí rozdílné poškození.

Pokud jednotka může vyhrát, vybere se cíl. Cíl je vybrán ze setu uložených jednotek, které jsou rozděleny do čtyř skupin: útočné budovy, útočné jednotky, dělníci a zbylé budovy. V tomto pořadí se určuje cíl, pokud je některý set prázdný, protože nepřítel nemá dané jednotky, jednoduše se přejde na následující set. Pokud je již vybraný cíl a není již zraněn, tak se rozhoduje o dalších jednotkách, které mohou být blíže, tudíž jsou lepší možností pro útok. Pokud je v blízkosti jednotka, která již utržila nějaké poškození a je v dosahu začne se útočit na ní. Dále Zergové disponují regeneraci zdraví. Toto je taktéž využito při útocích. Dílčí útok je rozdělen do tří stavů: může útočit, musí se stáhnout a pohyb na pozici. Jestliže má jednotka dostatečný počet životů může zaútočit na cíl. Avšak pokud jeho zdraví klesne na polovinu tak se stáhne a vyčkává, dokud se mu zdraví nevrátí nad polovinu celkového zdraví. Pokud jednotka není v dosahu cíle, přesune se na nepřítelovou pozici. Ukázka kódu 3 pro výběr cíle přiložena v příloze.

Jestliže však naše útočné síly nemohou vyhrát tak nastává situace, kdy se musí stáhnout. Únik je řešen dvěma způsoby. První způsob je vypočítání nové bezpečné pozice. Kdy s ohledem na pozici jednotky se vypočítá nová pozice, na kterou se jednotka přesune. Tato možnost je využita pokud není doposud vyzkoumána technologie „burrow“. Pokud je vyzkoumána tato technologie jednotky se zakopou. Problém při zakopání je ten, že jednotky se nemohou hýbat ani útočit. Pokud je jednotka odhalena tak se vykope a stahuje se za pomocí výpočtu nové pozice. Dále pokud je jednotka v dosahu útočících budov první co udělá je, že se stáhne mimo dosah a poté se může zakopat.

Pokud je jednotka zničena, zkontroluje se, zda jednotka patří nepřátelské nebo mé rase. Pokud je jednotka nepřátelská odstraní se ze setu, do kterého patří a vybírá se nový cíl pro útok. Dále se kontroluje v oblasti boje, zda je jednotka viditelná a tudíž může se počítat její poškození. Je zbytečné, pokud je jednotka mimo dosah, aby se s ní počítalo. Navíc jednotka mimo dosah je zakrytá ve fog of war a tudíž její atributy jsou pro bota skryté a můžou se vyskytovat chyby. Ukázka kódu 4 pro útok v příloze.

4.4 Testování a výsledky bota

Testování bota při vývoji programu bylo prováděno přímo ve hře StarCraft: Broodwar proti umělé inteligence, která je implementována ve hře. Bot, který je implementován ve hře využívá cheaty. Tedy, můžeme říct, že AI podvádí. Například počítač od začátku hry ví, na které z

výchozí pozice se nacházím. A navíc je komplexnější, aby mohl porazit přímo člověka, který proti němu nastoupí.

Proto jsem využil pro testování boty přímo ze soutěže SSCAIT, SSCAIT je studentský StarCraft AI turnaj, slouží jako konkurenční prostředí především pro studenty umělé inteligence a počítačových věd. Pravidla jsou striktně dané, tudíž žádné podvody zde nehrozí a oba boti mají stejné možnosti. Vybral jsem 3 boty různé rasy, kteří za sebou mají již zápasy v turnaji. Počet zápasů přesahuje stovky. V turnaji se nacházejí i nestudijní boti, proto jsem vybíral přímo studentské projekty pro testování. Jelikož jsem se nestihl přihlásit do soutěže na letošní rok, zvolil jsem tuhle strategii testování. Testování proběhlo stylem turnaje „každý s každým“, to znamená, že jsem v pěti vzájemných zápasech postavil vždy na stejné mapě všechny boty proti sobě. Dále jsem vybral tři mapy, na kterých se testování odehrávalo. Mapy se lišili především ve velikosti. První tabulka je pro mapu Destination, která je určená pro hru jeden na jednoho. Druhá tabulka je pro mapu Tau Cross, kde se nachází tři možné startovní lokace. Poslední tabulka je pro mapu Andromeda, kde jsou čtyři startovní lokace. Podle počtu startovních lokací se mění i velikost mapy. Veškeré testovací mapy jsou přímo ze soutěže, na kterých se odehrává turnaj.

Z následujících tabulek může vidět, jak můj bot obstál proti různým programátorům a rasám na různých mapách. Dále zde můžeme vidět i rozdíl mezi boty, které jsou ze soutěže SSCAIT. Můžeme zde také vidět rozdíl výsledků na různých velikostech mapy. Například bot rasy Protoss proti rase Zerg Na malé mapě Destination prohrál 4:1, ale na největší mapě Andromeda drtivě Zerga porazil. Toto je především tím, že čím větší mapa je, tak má bot větší šanci se proti early-game rase rozrůst a tím pádem získat převahu.

Taktiky a popisy botů protihráčů

Protoss

Tento bot je trochu hloupý. To především na menších mapách, kdy není schopný reagovat na early-rush. Dále také v hledání nepřítele a to ve smyslu, jakmile našel nepřítele a zničil mu nějakou budovu, zůstane v některých případech stát na místě a čeká, než přijde další jednotka, která objeví nový cíl. V nejhorším případě, co se většinou stávalo na největší mapě, byla situace, kdy mi vybil veškeré jednotky krom obraných, které ho následně zničili, ale bohužel jsem neměl minerály na naverbování jednotek a tím pádem nastala patová situace a ani já nebo on zůstal na své pozici. Bot rasy Protoss během testování použil dvě taktiky. První taktika byla jednoduchá a to rush s jednotkami Zealot, což je velmi účinná útočná jednotka, která hodně vydrží a udílí velké poškození. Proti této taktice jsem měl problémy a většinou prohrál. Další taktika byla využití jednotek Dark Templar, tato jednotka je účinná a její největší výhoda je, že je neviditelná. K těmto jednotkám přidal jednotku Dragon, která má taktéž vysoké poškození a útočí na dálku. Na první mapě, která je nejmenší jsem jasně dominoval a většinou nenechal protihráče rozrůst. Další mapa Tau Cross, která je o něco větší, srovnal tempo a výsledek již nebyl tak jednoznačný. Poslední a největší mapa byla v jeho režii. Na této mapě využíval především taktiku rushe se Ze-

aloty, což mě hned na začátku dostalo do defenzivní pozice a z ní jsem se již nedokázal nic udělat.

Terran

Bot téhle rasy je velmi pokročile naprogramovaný. Od začátků prozkoumává mapu a hledá si další místa pro těžbu minerálů. Během chvíle se rozroste po celé mapě. Útočné jednotky se skládají především s mariňáků. Tyto jednotky jsou silné, ale nebyl by problém, kdyby neměl Mediky, což je jednotka, která dokáže léčit své vlastní jednotky. Toto byl největší problém, protože jsem nedokázal včas zabíjet útočné jednotky, které disponují poměrně výbornou palebnou silou. Toto se také odrazilo na výsledcích. Další obrovská výhoda je použití dronů při obraně. Průměrný počet dronů na zápas se většinou pohybuje kolem 45 dronů. Což je velmi velké číslo a při obraně se hodí každá pomoc. Jediné vítězství bylo, když bot této rasy uznal porážku, díky překročení deseti minut zápasu.

Zerg

Tento bot je rovněž velmi dobře naprogramovaný, dokonce využívá učení. Bot si po konci zápasu ukládá výsledky a následně v dalších hrách je využívá ke změně strategie. Začínal s Four Pool, což je taktika čtyř dronů na těžbu, následného postavení budovy spawning pool a využití útočných jednotek Zergling k neustálému rushování protihráče. Na malé mapě rozhodovalo především počet jednotek a na mojí straně výborná obrana vlastní základny. Bohužel jakmile využil lepší taktiku tak jsem již neměl šanci. Dále na větších mapách vznikaly remízy, ke kterým docházelo především tím, že jsem nebyl schopný dobít já jeho, nebo on mě. Když se mu podařilo zničit mé těžební jednotky, tak nebyl schopný doničit zbytek základny a to díky obraným budovám na mojí straně. Remíza také nastala, pokud čas bitvy přesáhl deseti minut.

Název bota	můjBot	Terran	Protoss	Zerg	Celkově	Body	Pořadí
můjBot	X	0:5	5:1	2:3	7:9	21	3.
Terran	5:0	X	4:1	3:2	12:3	36	1.
Protoss	1:5	1:4	X	1:4	3:13	9	4.
Zerg	3:2	2:3	4:1	X	9:6	27	2.

Tabulka 3: mapa Destination

Název bota	můjBot	Terran	Protoss	Zerg	Celkově	Body	Pořadí
můjBot	X	1:4	3:3	4:5	8:12	24	3.-4.
Terran	5:1	X	3:2	2:3	10:6	30	1.-2.
Protoss	3:3	2:3	X	3:2	8:8	24	3.-4.
Zerg	5:4	3:2	2:3	X	10:9	30	1.-2.

Tabulka 4: mapa Tau Gross

Název bota	můjBot	Terran	Protoss	Zerg	Celkově	Body	Pořadí
můjBot	X	0:5	2:4	4:5	6:14	18	4.
Terran	5:0	X	6:4	3.2	14:6	42	1.
Protoss	4:2	4:6	X	4:1	12:9	36	2.
Zerg	5:4	2:3	1:4	X	8:11	24	3.

Tabulka 5: mapaAndromeda

5 Závěr

Hlavním cílem této bakalářské práce bylo naprogramovat a implementovat AI bota do hry StarCraft: Broodwar. Tímto úkolem jsem se dozvěděl různé věci nejen ohledně samotné hry StarCraft, ale také o různých AI metodách. Průzkumem těchto metodik jsem se dozvěděl užitečné informace, jak fungují dnešní AI v hrách. Vytvoření bota, který samostatně myslí a rozhoduje se, proběhlo úspěšně. Bot samostatně verbuje jednotky pro těžbu, nebo pro útok. Dále se také stará o výstavbu budov. Bojová stránka bota se stará o skautování a po následném nalezení nepřítele i o samostatný souboj.

Podle nových zkušeností s programováním AI a nastudovaných materiálů o tvorbě AI můžu usoudit, že vytvoření umělé inteligence pro hry je vskutku složitá. Dokonce i vytvoření umělé inteligence pro tak uzavřené prostředí jako je hra StarCraft je velice náročné. Tuto složitost nám především určují nedefinované stavy, které jsou ve hře, jako například fog of war, kdy máme zakrytou mapu a neznáme množství jednotek, nebo jejich pohyb.

Díky složitosti je bot rozdělen do hlavních odvětví. První je starost jednoho agenta o těžbu minerálů a verbování jednotek. Druhý agent má na starost všechny funkce, které se zabývají skautováním a soubojem. I nepatrné chyby u stavby budov, nebo verbování jednotek může vést ke zpoždění a následné porážce.

K testování bota byla využita již vytvořená inteligence hry StarCraft, která je zde implementována a vybraní boti přímo ze soutěže SSCAIT. Z výsledku je patrné, že můj bot na malé mapě, kde nepřítel je objeven rychle má větší šance na vítězství než na rozsáhlejších mapách. Dále pokud stojí proti stejné rase, šance na vítězství jsou padesáti procentní. Navíc ve většině případů nastala remíza. Největší nevýhoda a neúspěch byl proti botovi rasy Terran, který byl celkově velmi úspěšný i proti ostatním rasám. Proti rase Protoss můj bot obstál pouze na malých mapách, na větší mapě byl bez šance, protože nepřítel měl dost času na růst a zkoumání technologií.

Letos jsem bohužel přihlášení do soutěže nestihl a tak do budoucna plánuji vylepšení bota a nasazení do soutěže, kde bude bojovat proti AI botům dalších různých programátorů a výsledky se budou ukládat. Podle výsledku si pak ověřit funkčnost a preciznost mého bota.

Literatura

[1] *Starcraft Wiki[online]*, <starcraft.wikia.com/wiki/StarCraft_Wiki>

[2] RUSSEl, Stuart J a Peter NORVIG. *Artificial Intelligence: a modern approach. 3rd ed.*, Person new international ed. Harlow: Pearson, c2014. ISBN 878-1-292-02420-2.

[3] ROBERTSON, Glen a WATSON, Ian. *A Review of Real-Time Strategy Game AI*, New Zeland: University of Auckland.

[4] SYNNAEVE, Gabriel a BESSIERE, Pierre. *A Bayesian Model for RTS Units Control applied to StarCraft*, Seoul, South Korea, 2011

[5] ZHE, Wang, NGUYEN, Kien Quang, THAWONMAS, Ruck a RINALDO, Frank. *Using Monte-Carlo Planning fro Micro-Management in Starcraft*, Intelligent Computer Entertainment Laboratory, Ritsumeikan University, Kyoto, 2012

[6] *Úvod do neuronových sítí [online]*,
<http://www.statsoft.cz/file1/PDF/newsletter/2013_02_05_StatSoft_Neuronove_site_linky.pdf>

[7] BUCKLAND, Mat a Mark COLLINS. *AI Techniques for Game Programming.* United States of America: Premier Press, 2002. ISBN 1-931841-08-X.

[8] WEBER, Ben G, MATEAS, Michael a JHALA, Arnav. *Learning from Demonstration for Goal-Driven Autonomy*, UC Santa Cruz.

[9] *BWAPI Documentation[online]*, <https://bwapi.github.io/>

[10] PRATA, Stephen. *Mistrovství v C++*3. aktualiz. vyd. Přeložil Boris SOKOL. Brno: Computer Press, 2007. ISBN 978-80-251-1749-1

A Obsah CD

CD obsahuje jak hlavičkové soubory s příponou .h uložených ve složce Heads na kořenovém adresáři CD. Dále CD obsahuje složku Source, ve které se nacházejí hlavní soubory obsahující samotnou implementaci AI robota s příponou .cpp. Dále je zde .rar soubor, který obsahuje solution se všemi knihovnami BWAPI. V tomto souboru je přiložen .txt soubor, ve kterém je návod ke spuštění.
Obsah složky Source:

- AttackAgent.cpp - obsahuje třídu, která reprezentuje Attack agenta zodpovědného veškerý souboj a obranu proti nepříteli. Například výpočet možného vítězství s danýma jednotkami v určité lokaci, nebo kontrola a zvolení nové jednotky, na kterou se má útočit.
- AttackUnit.cpp - třída, ve které jsou obsaženy veškeré útočné jednotky, dále zde jsou funkce pro práci s těmito jednotkami.
- Buildings.cpp - třída, která je zodpovědná za stavbu budov a obsahuje informace o daných budov jako je čas kdy se začla stavět.
- ConstructAgent.cpp - tato třída obsahuje agenta, pro rozhodování jaké budovy se mají stavět, dále slouží pro kontrolu již postavených, nebo rozestavěných budov.
- ExampleAIModule.cpp - třída která již byla implementována v rámci BWAPI. Slouží jako základní třída, která je spuštěna na začátku programu. V této třídě se vytváří instance tříd MapInfo a MainAgent.
- MainAgent.cpp - třída, která obsahuje agenta ve kterém je hlavní loop, který nám rozhoduje, co vše se v dané iteraci bude dít. Dále se zde vytváří instance tříd UnitAgent, AttackAgeent, ConstructAgent.
- MapInfo.cpp - třída, ve které jsou uloženy informace o dané mapě, jako například: všechny možné výchozí pozice nepřítele, informace o mé vlastní pozici a po odhalení nepřítele i jeho pozice.
- UnitAgent.cpp - třída implementujícího agenta zodpovědného za verbování a správu jednotek pro těžbu, nebo pro správu populace.
- UpdateBuilding.cpp - tato třída se stará o funkce, které slouží k vývoji budov do další úrovně, nebo pro vývoj technologií.
- Worker.cpp - třída sloužící pro správu jednotek na těžbu, obsahuje informace o jednotkách, které těží minerály a plyny.

B Ukázky kódu

```
//for all units
for (auto &a : Broodwar->self()->getUnits())
{
   //check if unit is larva
   if (!a->getLarva().empty())
   {
      //is need worker
      if (unitAgent->isNeedWorker())
      {
            //make order to morph worker
            unitAgent->morphWorker(a);
      }
      //is need Overlord and spawning pool is build
      if (unitAgent->isNeedOverlord() && constructAgent->spawningPool)
      {
         unitAgent->morphToOverlord(a);
         //if morphing Overlord stop building
         if (!unitAgent->isMorphingOvelord)
            stopBuilding = true;
      }
      //is need Zergling
      if (unitAgent->isNeedZerglings())
      {
         unitAgent->morphToZergling(a);
      }
   }
}
```

Výpis 1: Verbování jednotek

```
if (attackUnit->getNumOfScoutUnits() == 6)
{
   //enemy not found
   if (!enemyFound)
   {
      if (!firstPos)
      {
         attackUnit->moveToNextPosition(nextEnemyPos);
         firstPos = true;
      }
      else
      {
         if (attackUnit->isInPosition())
         {
            nextEnemyPos = mapInfo->getNextStartPosition();
            attackUnit->moveToNextPosition(nextEnemyPos);
         }
      }
   }
   else
   {
      attackAgent->attackOnTarget();
   }
}
```

Výpis 2: Skautování mapy

```
void AttackAgent::selectTarget()
{
if (!targetSelected)
   {
      if (enemyAttackBuildingSet.size() > 0)
      {
         //select from attack buildings
         enemyTarget = selectFromAttackBuilding();
      }
      else
      {
         if (enemyAttackUnitSet.size() > 0)
         {
            //select from attack units
            enemyTarget = selectFromAttackUnits();
         }
         else
         {
            if (enemyWorkersSet.size() > 0)
            {
               //select from workers
               enemyTarget = selectFromWorkers();
            }
            else
            {
               if (enemyBuildingsSet.size() > 0)
               {
                  //select from other buildings
                  enemyTarget = selectFromBuildings();
               }
            }
         }
      }
   }
}
```

Výpis 3: Metoda selectTarget()

```
void AttackAgent::attackOnTarget()
{
   if (canWin())
   {
      //if units are buroow then unborrow
      attackUnit->unburrowUnits();
      //select target
      selectTarget();
      //if target is select then attack
      if (targetSelected)
         attackUnit->attack(enemyTarget);
   }
   else
   {
      if (burrowIsReady)
      {
         //burrow unit
         attackUnit->burrowUnits(enemyAttackBuildingSet, mapInfo->enemyPosition)
             ;
      }
      else
      {
         //go back
         attackUnit->moveBack(mapInfo->enemyPosition);
      }
   }
}
```

Výpis 4: Metoda attackOnTarget()